REMARKS

Abstract. The undersigned has amended the Abstract as requested by the Examiner.

Argument. Applicant wishes to thank the examiner for the care given to examining the claims. The applicant respectfully traverses the manner in which claim features are matched with elements disclosed in Gupta. Careful analysis of the Gupta system shows that Gupta is directed to another problem and solves this in a different manner, and that the elements disclosed in Gupta are neither similar to claimed elements, nor can they be brought into the functional relationship described in the claims.

In view of the fundamental differences between the cited prior art that require explanation, and the number of claims, the argumentation which follows is rather long. It has therefore been structured in the following manner:

1. Fundamental concepts regarding transactional behavior are summarized.
2. A brief summary of the essence of the present invention is given.
3. The Gupta system and the fundamental differences to the present invention are described.
4. The independent claims are reviewed, for each claim showing the differences from the Gupta disclosure in detail.

1. Fundamentals regarding "transactional behaviour"

Transactions and transactional behavior are at the core of the present invention, and a number of differences between the prior art and the present claims lies in the fact that the processes or services offered according to the invention exhibit transactional behavior.

This means that an invocation within a transaction will carry some token that identifies the transaction of the calling service. This is the transaction context. So, a service that receives an invocation from another service can detect any existing transaction by the presence of this token.

Consequently, the services according to the invention are aware of transactions and transaction tokens. When a service receives an invocation with a transaction token, it will wait for globalAbort/globalCommit messages associated with that same transaction. How these messages are sent is irrelevant, the important fact is that globalAbort/globalCommit must be accepted by the service, and the service must understand incoming transaction contexts. This makes the service definitely transaction-aware.

A transactional service supports undo or compensation operations by storing compensating operations for the event of a transaction being aborted (globalAbort). More on this will be said below, when comparing the invention to the Gupta system.

2. Summary of the present invention

The invention solves the problem of handling distributed transactions. The underlying issues are discussed in paragraphs [0012-0016]. One of the aims of the invention is to eliminate the need for a central transaction manager [0016, lines 1-3] and instead provide a mechanism by which completely autonomous components may interact transactionally without centralised coordination [0051].

This is done by using only local knowledge at each component [0052] and by ensuring that specific knowledge used for this is passed to the participating processes.

3. Summary of the Gupta patent and main differences to inventive system

Gupta discloses a transaction server that provides for transactions in which separate applications are integrated. The applications are independent and, more importantly, incompatible (e.g. col. 4, line 66 –

col. 5, line 2). The purpose of the Gupta system is to allow these incompatible applications to provide transactional behavior. However, the transactional behavior is implemented by the server, on behalf of the client applications, since the applications themselves do NOT provide for transactional behavior (column 6, lines 53-56).

- According to the invention, the applications taking part in transaction operations must be compatible in order to interact properly. They must at least understand the common transaction context that is passed around (see e.g. claim 23).

The system of Gupta is thus based on a centralized server to manage transactions (column 5, lines 13-15).

- According to the invention, the opposite is the case: there is no central server, but rather a network of peers where transactions are managed dynamically. The invention describes how such a peer operates, and what information needs to be exchanged between peers.

As an example, see e.g. claims 23 and further: a peer receives a transaction context, and accepts globalCommit and globalAbort notifications. In case of globalAbort, undo operations are scheduled by the peer. This is again different from Gupta, where undo operations are scheduled by the invoking coordinator, and the individual application does not offer undo operations.

Gupta assumes that the applications are unaware of any transactional behaviour (column 6, lines 53-56).

- According to the invention, the opposite is the case: applications also expose transactional behaviour in the form of transaction termination operations 'globalCommit' and 'globalAbort'. See, for instance, claim 11.

Consequently, Gupta does not define nor assume or claim any transactional co-operation between applications.

- According to the invention, the opposite is the case: transactional co-operation takes place between the applications themselves, and it is exactly this mode of co-operation that allows distributed applications to interact in a globally consistent transactional manner. See the globalCommit/globalAbort messages outlined in various claims.

Gupta is inherently based on comparing object states during transaction execution (an approach called

'optimistic locking', used in products like Hibernate etc.).

- According to the invention, no such locking mechanism is used. Instead, the states exchanged are counts, and this is done only _after_ transaction execution (i.e. during transaction termination). See, for instance, claim 11, the last three paragraphs.

3.1. Implementation of compensating operations

A service according to the **invention** receives an invocation and maintains its own undo information. All it exposes is the generic globalAbort to trigger this undo. All state information and algorithms for undo are kept hidden inside the service.

This is unlike **Gupta**, where the undo information and state is maintained in the central server. In Gupta, undo is no different from normal logic for the application, and the application doesn't even need to know whether it is being 'undone'. This is synonymous with the fact that Gupta assumes explicitly that applications are not transaction-aware and treat compensation as a regular business operation (column 11, lines 30-32).

The **invention** assumes the opposite: compensation is not a regular business operation but rather exposed as a transactional globalCommit/globalAbort transaction API where the exact business name of the operation is hidden for the outside. In other words: in **Gupta**, compensation of a 'buy' operation would be done by requesting a 'sell' operation on the same application.

In the **invention**, compensation of a 'buy' operation would be done by requesting a 'globalAbort' of the corresponding transaction id by the application. This is possible because the invention has a transaction API for the application, whereas Gupta doesn't. Consequently, the invention doesn't need similar compensation parameters like the object, verb/operation and 'Value' in Gupta's figure 3a.

3.2. A clarifying example: suppose you want to buy stock (shares) online.

- According to the **invention**, you would access an online broker service and buy shares with a transaction identifier. The broker service would know that you want to either confirm or undo at a later time, and waits for this to happen. If you want to undo, you send a globalAbort for the purchase, indicating what transaction you mean. The broker service is responsible for finding out how to undo that transaction, which stock you mean, etcetera.

- In **Gupta's** model, the stock broker service would offer two normal business operations: buy and sell. Once you buy, it is your responsibility to undo by constructing a sell invocation with the right stock, the amount to sell, etcetera. In addition, you may have to sell at a lower price than you bought. It is all up to you, the stock broker doesn't care, and doesn't expose any transactional functionality to do it.

3.3. Use of the term "process"

Please note that Gupta and the present invention make different use of the term "process":

- In Gupta, a process is a transaction involving one or more applications (col. 5, lines 11-15), where applications are independent and incompatible business applications (col. 5, lines 1-2). Several processes form a collaboration (col. 5, lines 11-12). Thus, a process according to Gupta is an activity, i.e. a transaction that is started, executed and then is stopped again after completion.

- According to the invention, e.g. as specified in claim 6, a process has an interface and implements a service defined by that interface. The process is permanently present, and the invocation of the service may cause a transaction to be executed. The same process may execute several transactions and ensures that these transactions do not conflict. The transactions are invoked by other (parent) transactions and may in turn invoke further (child) transactions in other processes (claim 6)

Thus, in **Gupta**, a process is a transaction, but according to the **invention** a process is the computer technical entity that executes a transaction or a local part of a global transaction.

If the term "process" is interpreted according to Gupta as being a transaction, then the phrases (e.g. from claim 6)

each process having an interface and implementing at least one respective service defined by that interface; a first invocation of the at least one respective service by a transaction resulting in the creation of a first transaction local to the process thereof,

[...]

each process characterized in that if the first transaction and the second transaction conflict but are both children of a same invoking transaction, then the first transaction and the second transaction are not executed concurrently;

each process further characterized in that each transaction local thereto is independently handled at the process;

each process making scheduling and recovery decisions independent of any centralized component.

would have to be read and interpreted as (replacing "*process*" by "transaction")

each *transaction* having an interface and implementing at least one respective service defined by that interface; a first invocation of the at least one respective service by a transaction resulting in the creation of a first transaction local to the *transaction* thereof,

[...]

each *transaction* characterized in that if the first transaction and the second transaction conflict but are both children of a same invoking transaction, then the first transaction and the second transaction are not executed concurrently;

each *transaction* further characterized in that each transaction local thereto is independently handled at the *transaction*;

each *transaction* making scheduling and recovery decisions independent of any centralized component.

which simply cannot be found to make any sense. Therefore, **this interpretation is not applicable** to the present invention.

On the other hand, if the interpretation of "process" according to the invention is used, then a number of inconsistencies arises, as shall be shown further below, in the context of the individual claims.

4. Present application

Claim 6

In addition to the fundamental differences already pointed out, **Gupta does not disclose**

> each process characterized in that if the first transaction and the second transaction conflict but are both children of a same invoking transaction, then the first transaction and the second transaction are not executed concurrently.

This method step involves transactions and how to maintain consistency when the same invoking transaction causes a conflict in its children. The rationale behind this solution is explained in [0065-0067].

Gupta, in the passage on col. 8, lines 44-51 (as cited in the office action) describes how subtransactions are invoked. There is no indication that certain transactions should not be executed concurrently.

The following paragraphs, col. 8, lines 52-67, show how a certain kind of conflict is detected: it is detected whether "no other transaction has modified this object between the last time the executing transaction modified it and the present time." If this is the case, an isolation fault is detected and a recovery service compensates certain transactions.

This is completely different from the inventive method and provides no hint of the inventive method, where transactions that are children of the same invoking transaction are involved, and where the solution is to not execute them concurrently (as opposed to generating a fault and triggering compensating transactions). **Gupta gives no hint whatsoever at this solution** to the problem outlined in [0065-0067].

Gupta further does not disclose

> each process further characterized in that each transaction local thereto is independently handled at the process;
>
> each process making scheduling and recovery decisions independent of any centralized component.

since Gupta in the passage on col. 9, lines 6-17 (as cited in the office action) shows the recording of transactions and subtransactions in a saga for the eventual case of undoing them (col. 9, lines 48-50, as

cited). This may be interpreted to mean that the transactions are stored and undone independently, but **it cannot be interpreted** (not even by hindsight) **to mean that there are independent processes that handles and undoes transactions**. For this, please recall that the term "process" must be interpreted consistently throughout the claim, and that a process according to claim 6 is an entity that provides a service, which service may be invoked to perform a transaction. It is these processes that are claimed to operate independent from one another.

In further contrast to the invention Gupta discloses a single interchange transaction server (Fig. 1, ref. no. 100) that handles all transactions centrally.

Therefore, the subject matter of claim 6 is not disclosed by Gupta. Neither Gupta nor the remaining prior art disclose or suggest the features of the last three paragraphs of claim 6, specifying the nature of each process.

Claim 8

The arguments set forth with regard to claim 6 are valid for claim 8 as well. Furthermore, Gupta does not disclose

> propagating from a first process to a second process a message indicative of a globalCommit operation with respect to a root transaction, said message also indicative of a number or identifying list of invocations which the first process has made to the second process on behalf of the root transaction;
>
> within the second process, comparing the number or list indicated in the message with a count or list within the second process of the number or list of invocations which have been made on behalf of the root transaction;
>
> in the event the comparison yields a non-match, aborting the transaction.

The Gupta passages cited with respect to these features, and the further passage on col. 8, lines 44-67 describe

- storing compensating records for subtransactions, for the event of a later rollback;
- checking whether all subtransactions have been processed;
- checking whether a business object that has been previously modified by the transaction has in

the mean time been modified by another transaction, and identifying an isolation fault if this is the case.

However, Gupta fails to disclose or suggest a number or identifying list of invocations being propagated from a first process to a second process and comparing said number or list with a number or list within the second process and aborting the transaction in the event of a non-match. Gupta does not compare lists or numbers of invocations, but processes a list of subtransactions and checks the status of business objects.

Therefore, the subject matter of claim 8 is not disclosed by Gupta. Neither Gupta nor the remaining prior art disclose or suggest the features of the last five paragraphs of claim 8, specifying the nature of each process.

Claims 10 and 11

Claims 10 and 11 differ in the last paragraph, covering the complementary alternatives resulting from the list comparison. The same arguments as for claims 6 and 8 apply here as well.

Claim 12 and claim 15

Gupta does not disclose that globalCommit messages carry information about the actual work being committed.

Gupta shows computer system level information being exchanged, i.e. the state of business software objects, and the transactions operating on these objects, in order to allow undo operations. This information is understood and used only by the system itself, and is not used outside the system.

According to the invention, information *about the actual work being committed* is distributed, in order to aid in resolving failures once they have occurred. This information is only passed along and stored by the system, as long as everything functions normally. Only in the cause of a failure is this information used, at a higher level, e.g. e.g. by a human, in order to "reorder the parts of the broken puzzle". This is made more explicit in system claims 13 and 14, and corresponding method claims 16 and 17.

For example, the system level information that is, according to the state of the art, returned by a failed transaction, may be something like "*there has been a system failure with bank X*". This is not helpful at all.

Thus, according to the invention, the information about the actual work such as "*payment of customer xyz with credit card No. 123 at bank X has not been completed, although the order has been confirmed* ..." is also available and can be used to clean up the remaining inconsistencies, which would be much more difficult without this information. During ordinary operation of the transaction system, this information is not used, and the transaction system will work just the same if this information is missing.

Gupta does not consider the problem of recovery after a failure by passing around and using information about the actual work being committed. Thus, claim 12 is neither disclosed nor suggested by Gupta.

Claim 18

Gupta does not disclose that (emphasis added)

> ... the propagated concurrency preferences at any level in the root invocation's invocation hierarchy specify the extent to which shared resource access is desired or allowed or denied among descendant invocations of the root invocation or user and other, concurrent invocations who are also descendants of the ***same*** root.

This states that the concurrency preferences are for descendants of the SAME root, which constitutes a difference with Gupta: Whereas Gupta's isolation levels take care of invocations among different transactions, according to the invention conflicts are addressed that are caused within the same overall (root) transaction.

Gupta does not consider the problem of conflicts occurring between different descendants of the same root transaction, and fails to provide the inventive solution. Thus, claim 18 is neither disclosed nor suggested by Gupta.

Claim 23

Claim 23 cannot be read consistently on the Gupta System: The "data management system, referred to as a service" might be read either

1. on the Gupta interchange server (100) or
2. on the Gupta applications (106).

In the first case, i.e. if the "service" is matched with the Gupta interchange server, then the "remote clients" must be matched with the Gupta applications. Then the claim features

> Some or all such remote clients having one or more associated contexts or transaction contexts;
>
> An invocation by a remote client also containing partial or complete information indicating or containing said client's context or contexts;

are not shown by Gupta, since the Gupta clients or applications are explicitly not aware of transactions and context (as already mentioned above, see column 6, lines 53-56).

In the second case, i.e. if the "service" is matched with the Gupta applications, then the claim features

> The service maintaining an undo operation for such a committed operation;
>
> A failing or failed remote client context leading to the execution of the undo operations of the corresponding committed invocations in the service.

are not shown by Gupta, since the Gupta clients, as above, are not aware of transactions and contexts, which is implied by providing commitment and related undo operations. According to Gupta, as explained earlier, such operations are handled by the central transaction server.

With regard to the claimed feature of "maintaining an undo operation" by the individual services, the explanation of differences, as given in the section "3.1. Implementation of compensating operations", applies.

Respectfully submitted,

/s/

Carl Oppedahl
PTO Reg. No. 32746
telephone 970 468 8600